An Artificial Immune System Approach With Secondary Response for Misbehavior Detection in Mobile *ad hoc* Networks

Slaviša Sarafijanović and Jean-Yves Le Boudec, *Fellow, IEEE*

***Abstract*—In mobile *ad hoc* networks, nodes act both as terminals and information relays, and they participate in a common routing protocol, such as dynamic source routing (DSR). The network is vulnerable to routing misbehavior, due to faulty or malicious nodes. Misbehavior detection systems aim at removing this vulnerability. In this paper, we investigate the use of an artificial immune system (AIS) to detect node misbehavior in a mobile *ad hoc* network using DSR. The system is inspired by the natural immune system (IS) of vertebrates. Our goal is to build a system that, like its natural counterpart, automatically learns, and detects new misbehavior. We describe our solution for the classification task of the AIS; it employs negative selection and clonal selection, the algorithms for learning and adaptation used by the natural IS. We define how we map the natural IS concepts such as self, antigen, and antibody to a mobile *ad hoc* network and give the resulting algorithm for classifying nodes as misbehaving. We implemented the system in the network simulator Glomosim; we present detection results and discuss how the system parameters affect the performance of primary and secondary response. Further steps will extend the design by using an analogy to the innate system, danger signal, and memory cells.**

***Index Terms*—Mobile, ad hoc, misbehavior, detection, artificial, immune, clonal selection, learning, adaptive.**

I. Introduction

A. Problem Statement: Detecting Misbehaving Nodes in DSR

MOBILE *ad hoc* networks are self organized networks without any infrastructure other than end-user terminals equipped with radios. Communication beyond the transmission range is made possible by having all nodes act both as terminals and information relays. This in turn requires that all nodes participate in a common routing protocol, such as dynamic source routing (DSR) [18]. A problem is that DSR works well only if all nodes execute the protocol correctly, which is difficult to guarantee in an open *ad hoc* environment.

A possible reason for node misbehavior is faulty software or hardware. In classical (non *ad hoc*) networks run by operators, equipment malfunction is known to be an important source of unavailability [19]. In an *ad hoc* network, where routing is performed by user provided equipment, we expect the problem to be exacerbated. Another reason for misbehavior stems from the desire to save battery power: some nodes may run a modified code that pretends to participate in DSR but, for example, does not forward packets. Finally, some nodes may also be truly malicious and attempt to bring the network down, as do Internet viruses and worms. An extensive list of such misbehavior is given in [3]. The main operation of DSR is described in Section II-A. In our simulation, we implement faulty nodes that, from time to time, do not forward data or route requests, or do not respond to route requests from their own cache.

We consider the problem of detecting nodes that do not correctly execute the DSR protocol. The actions taken after detecting that a node misbehaves range from forbidding to use the node as a relay [1] to excluding the node entirely from any participation in the network [3]. In this paper, we focus on the detection of misbehavior and do not discuss the details of actions taken after detection.

However, the actions do affect the detection function through the need for a secondary response. Indeed, after a node is disconnected (boycotted) because it was classified as misbehaving, it becomes nonobservable. Since the protection system is likely to be adaptive, the 'punishment" fades out and redemption is allowed [2]. As a result, a misbehaving node is likely to misbehave again, unless it is fixed, for example by a software upgrade. We call primary [respectively, secondary] response the classification of a node that misbehaves for the first [respectively second or more] time. Thus, we need to provide a secondary response that is much faster than primary response.

We chose DSR as a concrete example, because it is one of the protocols being considered for standardization for mobile *ad hoc* networks. There are other routing protocols, and there are parts of mobile *ad hoc* networks other than routing that need misbehavior detection, for example medium access control protocols. We believe the main elements of our method would also apply there, but a detailed analysis is for further work.

B. Traditional Misbehavior Detection Approaches

Traditional approaches to misbehavior detection [1], [3] use the knowledge of anticipated misbehavior patterns and detect them by looking for specific sequences of events. This is very efficient when the targeted misbehavior is known in advance (at system design) and powerful statistical algorithms can be used [4].

To detect misbehavior in DSR, Buchegger and Le Boudec use a reputation system [3]. Every node calculates the reputation of every other node using its own first-hand observations and the second-hand information obtained from others. The reputation of a node is used to determine whether countermeasures against

Manuscript received November 1, 2003; revised March 28, 2005. This work was supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under Grant 5005-67322.

The authors are with École Polytechnique Féderalé de Laussane (EPFL), Lausanne, Switzerland.

Digital Object Identifier 10.1109/TNN.2005.853419

the node are to be undertaken or not. A key aspect of the reputation system is how second-hand information is used, in order to avoid false accusations [3].

The countermeasures against a misbehaving node are aimed at isolating it, i.e., packets will not be sent over the node and packets sent from the node will be ignored. In this way, nodes are stimulated to cooperate in order to get service and maximize their utility, and the network also benefits from the cooperation.

Even if not presented by its authors as an artificial immune system (AIS), the reputation system in [3] and [4] is an example of a (nonbio inspired) immune system (IS). It contains interactions between its healthy elements (well-behaving nodes) and detection and exclusion reactions against nonhealthy elements (misbehaving nodes). We can compare it to the natural *innate* IS (Section II-B), in the sense that it is hardwired in the nodes and changes only with new versions of the protocol.

Traditional approaches miss the ability to learn about and adapt to new misbehavior. Every targeted misbehavior has to be imagined in advanced and explicitly addressed in the detection system. This is our motivation for using an AIS approach.

C. AIS Approaches

An AIS uses an analogy with the natural IS of vertebrates. As a first approximation, the IS can be described with the "self-nonself" model, as follows (we give more details in Section II-B).

The IS is thought to be able to classify cells that are present in the body as self and nonself cells. The IS is made of two distinct sets of components: the innate IS, and the adaptive IS. The innate IS is hard-wired to detect (and destroy) nonself cells that contain, or do not contain, specific patterns on their surface.

The adaptive IS is more complex. It produces a large number of randomly created detectors. A "negative selection" mechanism eliminates detectors that match any cell present in a protected environment (bone marrow and the thymus) where only self cells are assumed to be present. Noneliminated detectors become "naive" detectors; they die after some time, unless they match something (assumed to be a pathogen), in which case they become memory cells. Further, detectors that do match a pathogen are quickly multiplied ("clonal selection"); this is used to accelerate the response to further attacks. Also, since the clones are not exact replicates (they are mutated and the mutation rate is an increasing function of affinity between detectors and the pathogen) this provides a more focused response to the pathogen ("affinity maturation"). This also provides adaptation to a changing nonself environment.

The self-nonself model is only a very crude approximation of the adaptive IS. Another important aspect is the "danger signal" model [13], [14]. With this model, matching by the innate or adaptive mechanism is not sufficient to cause detection; an additional danger signal is required. The danger signal is, for example, generated by a cell that dies before being old. The danger signal model better explains how the IS adapts not only to a changing nonself, but also to some changes in self. There are many more aspects to the IS, some of which are not yet fully understood (see Section II-B).

D. AIS—Related Work

Hofmeyer and Forrest use an AIS for intrusion detection in wired local area networks [7], [8]. Their work is based on the negative selection part of the self-nonself model and some form of danger signal. In their system, transmission control protocol (TCP) connections play the role of self and nonself cells. TCP is a computer networking protocol that provides reliable data packets exchange between the two computers that communicate over a multihop computer network. One connection is represented by a triplet encoding the sender's destination address, the receiver's destination address, and the receiver's port number. A detector is a bit sequence of the same length as the triplet. A detector matches a triplet if both have M contiguous equal bits, where M is a fixed system parameter. Candidate detectors are generated randomly; in a learning phase, detectors that match any correct (i.e., self) triplets are eliminated. This is done offline, by presenting only correct TCP connections. Noneliminated detectors have a finite lifetime and die unless they match a nonself triplet, as in the IS. The danger signal is also used: it is sent by humans as confirmation in case of potential detection. This is a drawback, since human intervention is required to eliminate false positives, but it allows the system to learn changes in the self. With the terminology of statistical pattern classification, this use of the danger signal can be viewed as some form of supervised training. Similarly, Dasgupta and González [22] use an AIS approach to intrusion detection, based on negative selection and genetic algorithms.

A major difficulty in building an AIS in our framework is the mapping from biological concepts to computer network elements. Kim and Bentley [9] show that straightforward mappings have computational problems and lead to poor performance, and they introduce a more efficient representation of self and nonself than in [7]. They show the computational weakness of negative selection and add clonal selection to address this problem [9]. In their subsequent papers, they examine clonal selection with negative selection as an operator [10], and dynamical clonal selection [11], showing how different parameters affect detection results. For an overview of AIS, see [20] and [21].

In our previous work on building an AIS for misbehavior detection in mobile *ad hoc* networks [5] we implemented negative selection. Negative selection is used for learning about the protected system, but it does not provide adaptation to misbehavior. In this paper, we also implement adaptation, using clonal selection inspired by [10]. Clonal selection provides a faster secondary response to repeated misbehavior, as discussed in Section I-A.

E. Contribution of This Paper and Organization

Our long-term goal is to understand whether our previous work, based on the traditional approach [2], can benefit from an AIS approach that introduces learning and adapting mechanisms. In the DSR example, this means adding IS code to every node, so that it becomes resistant to other nodes' misbehavior.

The first problem to solve is mapping the natural IS concepts to our framework. This is a key issue that strongly influences the detection capabilities. We describe our solution in Section III-B. For the representation of self-nonself and for the matching functions, we start from the general structure proposed by Kim and Bentley [9], which we adapt to our case. Then we define the resulting algorithm, which is based on negative selection, clonal selection, and an *ad hoc* classification rule. Although negative selection is done like Hofmeyer and Forrest in [7] and [8], and clonal selection like Kim and Bentley in [9], the detection rule

is our original component aimed at decreasing false positives. Our other contributions are: 1) the definition of a mapping and construction of an AIS adapted to our concrete application, and 2) the AIS implementation in the Glomosim simulator [17] and its performance analysis. We examine and show a positive impact of clonal selection on the secondary response time. The response time is a new AIS performance dimension introduced by the detection rule we use. It is not present in the previously mentioned related works, though it is an important feature of the human IS response.

The paper is organized as follows. Section II gives background and terminology on DSR and the natural IS. Section III gives the mapping from the IS to the detection system for DSR misbehavior detection, and the detailed definition of the detection system. Section IV gives simulation specific assumptions and constraints, simulation results and discussion of the results. Section V draws conclusions and describes what we have learned and how we will exploit it in future steps.

II. Background

A. DSR: Basic Operations

DSR is one of the candidate standards for routing in mobile *ad hoc* networks [18]. A "source route" is a list of nodes that can be used as intermediate relays to reach a destination. It is written in the data packet header at the source. Intermediate relays simply look it up to determine the next hop.

DSR specifies how sources discover, maintain, and use source routes. To discover a source route, a node broadcasts a route request packet. Nodes that receive a route request add their own address in the source route collecting field of the packet and then broadcast the packet, except in two cases. The first case is if the same route request was already received by a node; then the node discards the packet. Two received route requests are considered to be the same if they belong to the same route discovery, which is identified by the same value of source, destination, and sequence number fields in the request packets. The second case is if the receiving node is destination of the route discovery, or if it already has a route to the destination in its cache; then the node sends a route reply message that contains a completed source route. If links in the network are bidirectional, the route replies are sent over the reversed collected routes. If links are not bidirectional, the route replies are sent to the initiator of the route discovery as included in a new route request generated by answering nodes. The new route requests will have the destination be the source of the initial route request. The node that initiates an original route request receives usually more route replies, each containing a different route. The replies that arrive earlier than others are expected to indicate better routes, because for a node to send a route reply, it is required to wait first for a time proportional to the number of hops in the route it has as the answer. If a node hears that some neighbor node answers during this waiting time, it supposes that the route it has is worse then the neighbor's one, and it does not answer. This avoids route reply storms and unnecessary overhead.

After the initiator of route discovery receives a first route reply, it sends data over the obtained route. While packets are sent over the route, the route is maintained, in such a way that every node on the route is responsible for the link over which it sends packets. If a link in the route breaks, the node that detects that it cannot send over that link should send error messages to the source. Additionally it should salvage the packets destined to the broken link, i.e., reroute them over alternate partial routes to the destination.

The mechanisms described previously are the basic operation of DSR. There are also some additional mechanisms, such as gratuitous route replies, caching routes from forwarded or overheard packets and DSR flow state extension [18].

B. Natural IS

The main function of the IS is to protect the body against different types of pathogens, such as viruses, bacteria, and parasites and to clear it from debris. It consists of a large number of different innate and acquired immune cells, which interact in order to provide detection and elimination of the attackers [15]. We present a short overview based on the self-nonself and the danger models [14], [15].

1) Functional Architecture of the IS: The first line of defense of the body consists of physical barriers: skin and mucous membranes of the digestive, respiratory, and reproductive tracts. It prevents the body from being entered easily by pathogens.

The innate IS is the second line of defense. It protects the body against common bacteria, worms, and some viruses, and clears it from debris. It also interacts with the adaptive IS, signaling the presence of damage in self cells and activating the adaptive IS.

The adaptive IS learns about invaders and tunes its detection mechanisms to better match previously unknown pathogens. It provides an effective protection against viruses even after they enter the body cells. It adapts to newly encountered viruses and memorizes them for more efficient and fast detection in the future.

2) Innate IS: Consists of macrophage cells, complement proteins, and natural killer cells. Macrophages are large cells that are attracted by bacteria to engulf the bacteria in the process called "phagocytosis". Complement proteins can also destroy some common bacteria. Both macrophages and complement proteins send signals to other immune cells when there is an attack.

3) Adaptive IS: Consists of two main types of lymphocyte cells. These are B cells and T cells. Both B and T cells are covered with antibodies. Antibodies are proteins capable of chemically binding to nonself antigens. Antigens are proteins that cover the surface of self and nonself cells. Whether chemical binding takes place between an antibody and an antigen depends on the complementarity of their three-dimensional (3-D) chemical structures. If it does, the antigen and the antibody are said to be "cognate." Because this complementarity does not have to be exact, an antibody may have several different cognate antigens. What happens after binding depends on additional control signals exchanged between different IS cells, as we explain next.

One B cell is covered by only one type of antibody, but two B cells may have very different antibodies. As there are many B cells (about 1 billion fresh cells are created daily by a healthy human), there are also a large number of different antibodies at the same time. How is this diversity of antibodies created and why do antibodies not match self antigens? The answer is in the

process of creating B cells. B cells are created from stem cells in the bone marrow by rearrangement of genes in immature B cells. Stem cells are generic cells from which all immune cells derive. Rearrangement of genes provides diversity of B cells. Before leaving bone marrow, B cells have to survive **negative selection**: if the antibodies of a B cell match any self antigen present in the bone marrow during this phase, the cell dies. The cells that survive are likely to be self tolerant.

B cells are not fully self tolerant, because not all self antigens are presented in bone marrow. **Self tolerance** is provided by T cells that are created in the same way as B cells, but in the thymus, the organ behind the breastbone. T cells are self tolerant because almost all self antigens are presented to these cells during negative selection in the thymus.

After some antibodies of a B cell match antigens of a pathogen or self cell (we call this event "signal 1b") that antigens are processed and presented on the surface of the B cell. For this, major-histocompatibility-complex (MHC) molecules are used and their only function. If antibodies of some T cell bind to these antigens and if the T cell is activated (by some additional control signal) the detection is verified and a confirmation signal sent from T cell to B cell (we call this event "signal 2b"). Signal 2b starts the process of producing new B cells, that will be able to match the pathogen better. This process is called clonal selection. If signal 2b is absent, it means that the detected antigens are probably self antigens for which the T cells are tolerant. In this last case, the B cell will die, together with its self reactive antibodies.

B cells can begin clonal selection without confirmation by signal 2b, but only in the case when matching between B cell antibodies and antigens is very strong. This occurs with a high probability only for memory B cells, the cells that were verified in the past to match nonself antigens.

In the **clonal selection** phase, a B cell divides into a number of clones with similar but not strictly identical antibodies. Statistically, some clones will match the pathogen that triggered the clonal selection better than the original B cells and some will match it worse. If the pathogens whose antigens triggered clonal selection are still present, they will continue to trigger cloning new B cells that match the pathogen well. The process continues reproducing B cells more and more specific to present pathogens. B cells that are specific enough become memory B cells and do not need **co-stimulation** by signal 2b. This is a process with positive feedback and it produces a large number of B cells specific to the presented pathogen. Additionally, B cells secrete chemicals that neutralize pathogens. The process stops when pathogens are cleared from the body. Debris produced by the process are cleared by the innate IS. Memory B cells live long and they are ready to react promptly to the same cognate pathogen in the future. Whereas first time encountered pathogens require a few weeks to be learned and cleared by the IS, the secondary reaction by memory B cells takes usually only a few days.

4) Clustering: Recognition of a pathogen by a B cell requires that the antibodies (receptors) of the B cell bind to not only one but more antigens that belong to the pathogen. This is called **clustering**. It provides some robustness to false antibody-antigen binding.

5) Danger Signal: Is an additional control used for activating T cells. After T cell antibodies bind to antigens presented by MHC of a B cell (signal 1t), the T cell is activated and sends signal 2b to a B cell only if it receives a confirmation signal (signal 2t) from an antigen presenting cell (APC). The APC will give signal 2t to a T cell only if it engulfed the same nonself antigens, which happens only when the APC receive a danger signal from self cells or from the innate IS. The danger signal is generated when there is some damage to self cells, which is usually due to pathogens. As an example, the danger signal is generated when a cell dies before being old; the cell debris are different when a cell dies of old age or when it is killed by a pathogen.

There are many other subtle mechanisms in the IS, and not all of them are fully understood. In particular, time constants of the regulation system (lifetime of B and T cells, probability of reproduction) seem to play an important role in the performance of the IS [16]. We expect that we have to tune similar parameters carefully in an AIS.

III. Design of Our Detection System

A. Overview of Detection System

Every DSR node implements an instance of the detection system, and runs it in two phases. In an initial phase, the detection system learns about the normal behavior of the nodes with respect to the DSR protocol. During this phase, the node is supposed to be in a protected environment in which all nodes behave well. From received or overheard packets, the node observes the behavior of its neighbors and represents it by the antigens (Section III-C). At the end of this learning phase, the node runs the negative selection process and creates its antibodies, which we call detectors (Section III-C). In general terms, the first phase implements a special form of supervised learning, where only positive cases are used for training.

After the learning is done, the node may leave the protected environment and enter the second phase where detection and classification are done. In this phase, the node may be exposed to misbehaving nodes. Detectors are used for checking if newly collected antigens represent the behavior of good or bad nodes. If an antigen, created for some neighbor during some time interval, is detected by any of the detectors, the neighbor is considered to be suspicious in that time interval. If there are relatively many suspicious intervals for a neighbor, that neighbor is classified as misbehaving (Section III-D). This triggers the clonal selection process (Section III-E) in the node that made the classification. In this process, the node adapts its detectors to better detect experienced misbehavior. This results in a better response if the same or similar misbehavior is encountered again. In general terms, the second phase implements a form of reenforcement learning.

The detailed algorithm is given in the Appendix.

B. Mapping of Natural IS Elements to Our Detection System

The elements of the natural IS used in our detection system are mapped as follows.

- Body: the entire mobile *ad hoc* network.
- Self Cells: well behaving nodes.

- Nonself Cells: misbehaving nodes.
- Antigen: sequence of observed DSR protocol events recognized in sequence of packet headers. Examples of events are "data packet sent," "data packet received," "data packet received followed by data packet sent," "route request packet received followed by route reply sent." The sequence is mapped to a compact representation as explained in Section III-C.
- Antibody: a pattern with the same format as the compact representation of antigen (Section III.C).
- Chemical Binding: binding of antibody to antigen is mapped to a "matching function," as explained in Section III-C.
- Bone Marrow: Antibodies are created during an offline learning phase. The bone marrow (protected environment) is mapped to a network with only certified nodes. In a deployed system, this would be a testbed with nodes deployed by an operator; in our simulation environment, this is a preliminary simulation phase.

C. Antigen, Antibody, and Negative Selection

1) Antigens: Antigens could be represented as traces of observed protocol events. However, even in low bit-rate networks, this rapidly generates sequences that are very long (for a 100-s observation interval, a single sequence may be up to 1-Gb long), thus, causing generation a large number of patterns prohibitive. This was recognized and analyzed by Kim and Bentley in [9] and we follow the conclusions, which we adapt to our case, as we describe now.

A node in our system monitors its neighbors and collects one protocol trace per monitored neighbor. A protocol trace consists of a series of data sets, collected on nonoverlapping intervals during the activity time of a monitored neighbor. One data set consists of events recorded during one time interval of duration Δt ($\Delta t = 10$ s by default), with an additional constraint to maximum N_s events per a data set ($N_s = 4\,0$ by default).

Data sets are then transformed as follows. First, protocol events are mapped to a finite set of primitives, identified with labels. In the simulation, we use the following list:

A=RREQ sent
B = RREP sent
C = RERR sent
D = DATA sent and IP source address is not of monitored node
E = RREQ received
F = RREP received
G = RERR received
H = DATA received and IP destination address is not of the monitored node

A data set is then represented as a sequence of labels from the alphabet defined previously, for example s

$$l_1 = (\texttt{EAFBHHEDEBHDHDHHDHD}, \ldots).$$

Second, a number of "genes" are defined. A gene is an atomic pattern used for matching. We use the following list:

Gene1 = #E in sequence
Gene2 = #(E $*$ (A **or** B)) in sequence
Gene3 = #H in sequence
Gene4 = #(H $*$ D) in sequence

where #("subpattern") is the number of the subpatterns "subpattern" in a sequence, with $*$ representing one arbitrary label or no label at all. For example, #[E$*$(A **or** B)] is the number of subpatterns that are two or three labels long, and that begin with E and end with A or B.

The genes are defined in such a way to extract only important features of the routing protocol to be included into the final antigen representation. For example, Gene2 gives some indication on how the observed node handles route requests. The genes are used to convert the very large space of all possible observation sequences (like l_1) into a space of the antigens' representations (like l_2, see the following) of a computationally feasible size. For example, l_1 is mapped to an antigen that consists of the following four genes:

$$l_2 = (3 \quad 2 \quad 7 \quad 6).$$

Moreover, the value of Gene2 is positively correlated with the value of Gene1, for the nodes that handle route requests correctly. Such dependencies between the genes are important part of the representation choice, as they enable differentiation between the normal and abnormal observed behavior cases.

We select and define the genes manually, but the process can be automated, as discussed in Section V.

Finally, a gene value is encoded on 10 b as follows. A range of the values of a gene, that are below some threshold value, is uniformly divided on 10 intervals. The position of the interval to whom the gene value belongs gives the position of the bit that is set to 1 for the gene in the final representation. The threshold is expected to be reached or exceeded rarely. The values above the threshold are encoded as if they belong to the last interval. Other bits are set to 0. For example, if the threshold value for all the four defined genes is equal to 20, l_2 is mapped to the final antigen format:

$$l_3 = (0\,000\,000\,010 \;\; 0\,000\,000\,010 \;\; 0\,000\,001\,000 \;\; 0\,000\,001\,000).$$

With such encoding, misbehavior of a node will change the positions of 1 s in the genes that participate in the representation of the corresponding behavior feature (handling route requests, for example). The positions will change more if the node misbehaves more, and similar behavior will have antigens with 1 s at similar positions. This is important for the implementation of the clonal selection: some of random mutations in the antibodies should refine them to better cover the misbehavior areas of the all-possible-antigens space.

There is one antigen such as l_3 every Δt seconds, for every monitored node, during the activity time of the monitored node. Every bit in this representation is called a "nucleotide."

2) Antibody and Matching Function: Antibodies have the same format as antigens (such as l_3), except that they may have any number of nucleotides equal to 1 (whereas an antigen has exactly one per gene). An antibody matches an antigen (i.e., they are cognate) if the antibody has a 1 in every position where the antigen has a 1. This is the same as in [10] and is advocated there as a method that allows a detection system to have good coverage of a large set of possible nonself antigens with a relatively small number of antibodies.

The genes are defined with the intention to translate raw protocol sequences into more meaningful descriptions.

3) Negative Selection: Antibodies are created randomly, uniformly over the set of possible antibodies. During the offline

learning phase, antibodies that match any self antigen are discarded (negative selection).

D. Node Detection and Classification

Matching an antigen is not enough to decide that the monitored node misbehaves, since we expect, as in any AIS, that false positives occur. Therefore, we make a distinction between detection and classification. We say that a monitored node is **detected** (or "suspicious") in one data set (i.e., in one interval of duration Δt) if the corresponding antigen is matching any antibody. Detection is done per interval of duration Δt (=10 s by default). A monitored node is **classified as "misbehaving"** if the probability that the node is suspicious, estimated over a sufficiently large number of data sets, is above a threshold. This is similar to the clustering of antigens matched by the antibodies of a B cell (Section II-B). The threshold is computed as follows.

Assume we have processed n data sets for this monitored node. Let M_n be the number of data sets (among n) for which this node is *detected* (i.e., is suspicious). Let $\theta_{\max}$ be a bound on the probability of false positive detection (detection of well behaving nodes, as if they are misbehaving) that we are willing to accept, i.e., we consider that a node that is detected with a probability $\leq \theta_{\max}$ is a correct node (we take by default $\theta_{\max} = 0.06$). Let α (=0.0001 by default) be the classification error that we we target. We classify the monitored node as misbehaving if

$$M_n > K(n) \tag{1}$$

where

$$K(n) = \begin{cases} \min\{k : \sum_{i=k+1}^{n} B_{n,\theta_{\max}}(k) \leq \alpha\} & \text{if } n\theta_{\max} \leq 5 \\ n\theta_{\max}\left(1 + \frac{\xi(\alpha)}{\sqrt{n}}\sqrt{\frac{1-\theta_{\max}}{\theta_{\max}}}\right) & \text{otherwise} \end{cases} \tag{2}$$

$B_{n,\theta_{\max}}$ is binomial distribution with parameters n and $\theta_{\max}$, and $\xi(\alpha)$ is the $(1-\alpha)$-quantile of the normal distribution (for example, $\xi(0.0001) = 3.72$). As long as (1) is not true, the node is classified as well behaving. For the default parameter values, $n\theta_{\max} \leq 5$ corresponds to $n \leq 83$. For $n \leq 83$, $K(n)$ is precalculated and used from a table (the table for the default value of α is given in the Appendix). For $n > 83$, the (1) and (2) are equivalent to $(M_n/n) > 0.06 + (0.88/\sqrt{n})$. The derivation of (2) is given in the Appendix.

E. Clonal Selection

As explained in Section II-B, the antibodies in the IS that are activated by both signal 1b and signal 2b are selected to undergo clonal selection. We define signals 1b and 2b in our setting as follows.

Signal 1b) When a node classifies a neighbor as misbehaving (Section III-D), signal 1b is received by a fraction W (by default 15%) of the the best scored detectors of that node. The scores are calculated as follows. Every Δt the observed antigens are exposed to the detectors of the node. Whenever a detector is matching an antigen, its score is increased by one. After the detectors that receive the signal 1b are selected, the scores of all the detectors are reset to zero.

Signal 2b) We do not implement an exact analog to signal 2b. This is because we do not model helper T-cells (these cells generate signal 2b in the IS) and their activation (Section II-B). Instead, we use a negative selection method, as in [10], defined later in this section, to control the clonal selection process once it starts. Passing the negative selection test by a mutated antibody may be seen as the equivalent of receiving some type of signal 2b, as it ensures self-tolerance. Indeed, one important function of helper T-cells is providing self-tolerance.

The clonal selection mechanism with negative selection as an operator works as follows. A fraction W of the best scored detectors of the node receive signal 1b and enter the clonal selection process, in which each of them will produce one new additional detector. Every detector that enters the process is cloned; the clone is mutated, and if it matches some of the self-antigens in the node's collection, it is eliminated (negative selection test). This is repeated until one mutated clone is generated that survives the negative selection test. Mutation is defined as random flipping bits of the detector, which occurs per bit with some small probability p (by default 0.1). The newly generated detectors are substituted to the same number of this node's detectors; the detectors that had the worse scores are eliminated.

For example, let d be a node's detector that is selected to undergo clonal selection. A possible clonal selection process scenario is as follows:

$$d = (1\,100\,010\,110\ 1\,110\,110\,010\ 0\,010\,100\,000\ 1\,000\,001\,101).$$

From the set (S) of self antigens that the node has collected during the learning phase, let a_1 and a_2 are two examples

$$a_1 = (0\,000\,000\,010\ 0\,000\,000\,010\ 0\,000\,001\,000\ 0\,000\,000\,100)$$
$$a_2 = (0\,000\,000\,010\ 0\,000\,000\,010\ 0\,000\,001\,000\ 0\,000\,001\,000).$$

We see that d does not match a_1 or a_2 (the matching rule is given in Section III-C.2); it does not match any of the antigens from S, by the rule how it is generated. Let d_1 be the first mutated clone of d, which happens to differ from d in 3 b

$$d_1 = (1\,100\,010\,110\ 1\,110\,110\,011\ 0\,010\,101\,000\ 1\,000\,001\,001).$$

Because d_1 matches a_2, d_1 is deleted and another mutated clone d_2 is created, which happens to differ from d in 4 b

$$d_2 = (1\,100\,010\,110\ 1\,100\,110\,010\ 0\,011\,100\,000\ 0\,000\,011\,101).$$

We see that d_2 does not match a_1 or a_2, and assuming that it does not match any other antigen from the set S, d_2 becomes a valid detector, and will substitute a badly scored one from the set S.

Because the length of the detectors is 40 b and the mutation probability per bit is 0.1, mutated clones will differ from the

TABLE I
DETECTION SYSTEM PARAMETERS

Parameters	Default values
maximal number of self antigens collected for learning	450
maximal time for collecting self antigens for learning	200 s
number of detectors	80
number of genes in an antigen	4
number of nucleotides per gene	10
max. number of protocol events recorded in a data set	40
maximal time for collecting a data set (an antigen)	10 s
accepted misbehavior threshold $\theta_{\max}$	0.06
targeted classification error ratio α	0.0001
percentage of detectors selected for clonal selection	0.15
probability of mutation per bit	0.1

original detectors from which they are created by 4 b on average (before negative selection). When detectors are created in the learning phase, they differ on average by 20 b before negative selection. This means that clonal selection increases the percentage of the detectors that are similar but slightly different to good scored ones. It is intuitively clear from these facts, and confirmed by the experiments, that the clonal selection improves the detection results.

F. System Parameters

In this implementation, the default values of parameters (Table I) are chosen from extensive pilot simulation runs, as a compromise between good detection results and a small memory and computation usage by the detection system.

IV. SIMULATION RESULTS

A. Description of Simulation

1) Experimental Setup: We have implemented our detection system in Glomosim [17], a simulation environment for mobile *ad hoc* networks. We use the simulator's DSR code and modify it only to allow nodes' misbehavior. The detection system code that we add can be run in two versions: with or without the clonal selection mechanism.

We simulate a network of 40 nodes. The simulation area is a rectangle with the dimensions of 800 m × 600 m. The nodes are initially deployed on a grid with the distance between neighbors equal to 100 m. The mobility model is a random way-point. The speed of nodes is a parameter, and we set it to be constant for one simulation run. The radio range is 380 m. Traffic is generated as constant bit-rate, with packets of length 512 B sent every 0.2–1 s.

We run two series of experiments.

- (Without clonal selection:) In the first set of experiments, clonal selection is disabled. The initial set of detectors generated during the learning phase is used unchanged during the detection and classification phase.
- (With clonal selection:) Then we run a second set of experiments, now with clonal selection. Every experiment consists of four consecutive phases, in order to obtain primary and secondary responses. The first phase is learning an initial set of detectors (as in the first experiment). The second phase is detection and classification, but now the detectors start to change and adapt to misbehavior during the phase. This allows us to measure the primary response metrics. In the third phase, nodes do not misbehave, and the system forgets about previous detections, but the set of detectors obtained at the end of the second phase is kept unchanged. The fourth phase is again a detection and classification phase, with the same misbehavior as in the second phase, but the initial set of detectors is now different (it is the set that is achieved at the end of the second phase). This gives the secondary response metrics. The conditions are otherwise the same as in the first set of experiments.

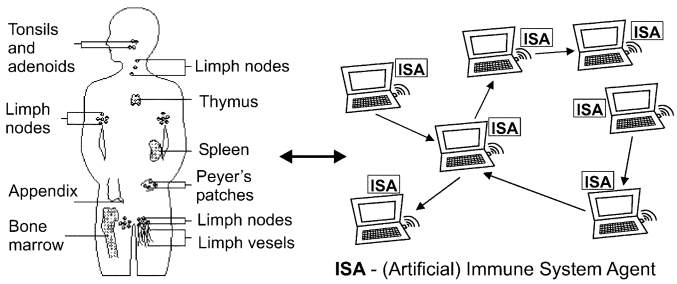


Fig. 1. From the natural IS to an AIS. Making DSR immune to node misbehavior.

We performed 20 independent replications of all experiments, in order to obtain good 90% confidence intervals.

For all the simulations, the parameters have default values (Table I), unless otherwise mentioned.

2) Misbehavior: Is implemented by modifying the DSR implementation in Glomosim. We implemented two types of misbehavior: 1) nonforwarding route requests and nonanswering from its route cache, and 2) nonforwarding data packets. A misbehaving node does not misbehave for every packet. In contrast, it does so with fixed probabilities, which are also simulation parameters (default values are 0.6). In our implementation, a node is able to misbehave with different probabilities for the two types of misbehavior. In the simulation, we always set both misbehavior probabilities to the same value (what is not necessary). The number of misbehaving nodes is also a simulation parameter (default value is 5).

3) Performance Metrics: We show simulation results with the following metrics:

Average Time until Correct Classification is the time until a given node, that is running the detection algorithm, classifies a given misbehaving node as misbehaving for the first time [after a sufficiently large number of positive detections occurred, see (1)], averaged over all pairs $\langle$ node i that is running the detection algorithm, node j that is classified as misbehaving by node $i\rangle$. When we talk about "**response time**" of the detection system, we refer to this metric.

True Positive Classification Ratio is the percentage of misbehaving nodes that are classified as misbehaving.

False Positive Classification Ratio is the percentage of well behaving nodes that are mistakenly classified as misbehaving.

B. Simulation Results

1) Classification Capabilities: For all simulation runs, all misbehaving nodes are detected and classified as misbehaving by all their neighbors in the static case and by all nodes in the cases with mobility. The main effect on other classification

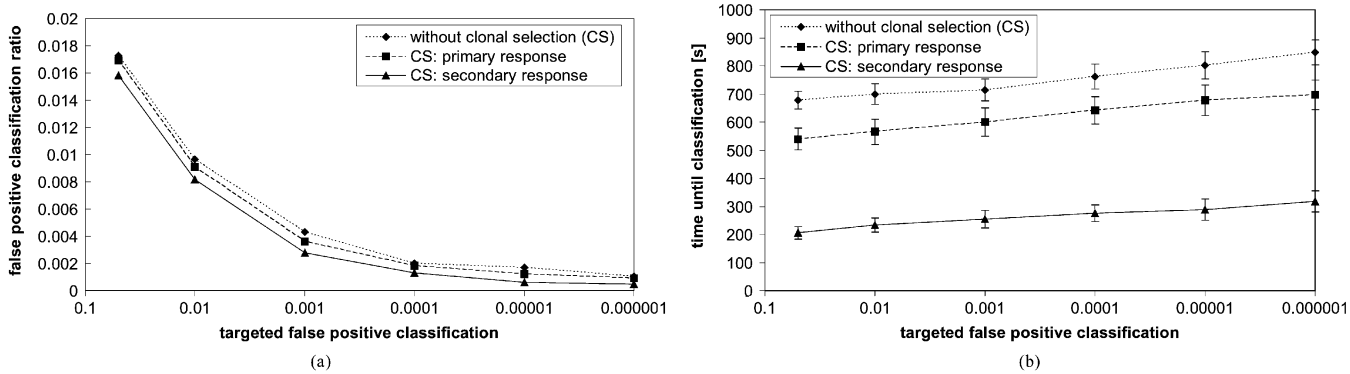


Fig. 2. AIS main metrics. (a) Average ratio of misclassification for well behaving nodes (false positive). (b) Average time until correct classification for misbehaving nodes versus target false positive classification probability α, for tree cases: without clonal selection (CS), with CS-primary response, with CS-secondary response. Comment: true positive classification probability is equal to 1, which is not shown on the graphs.

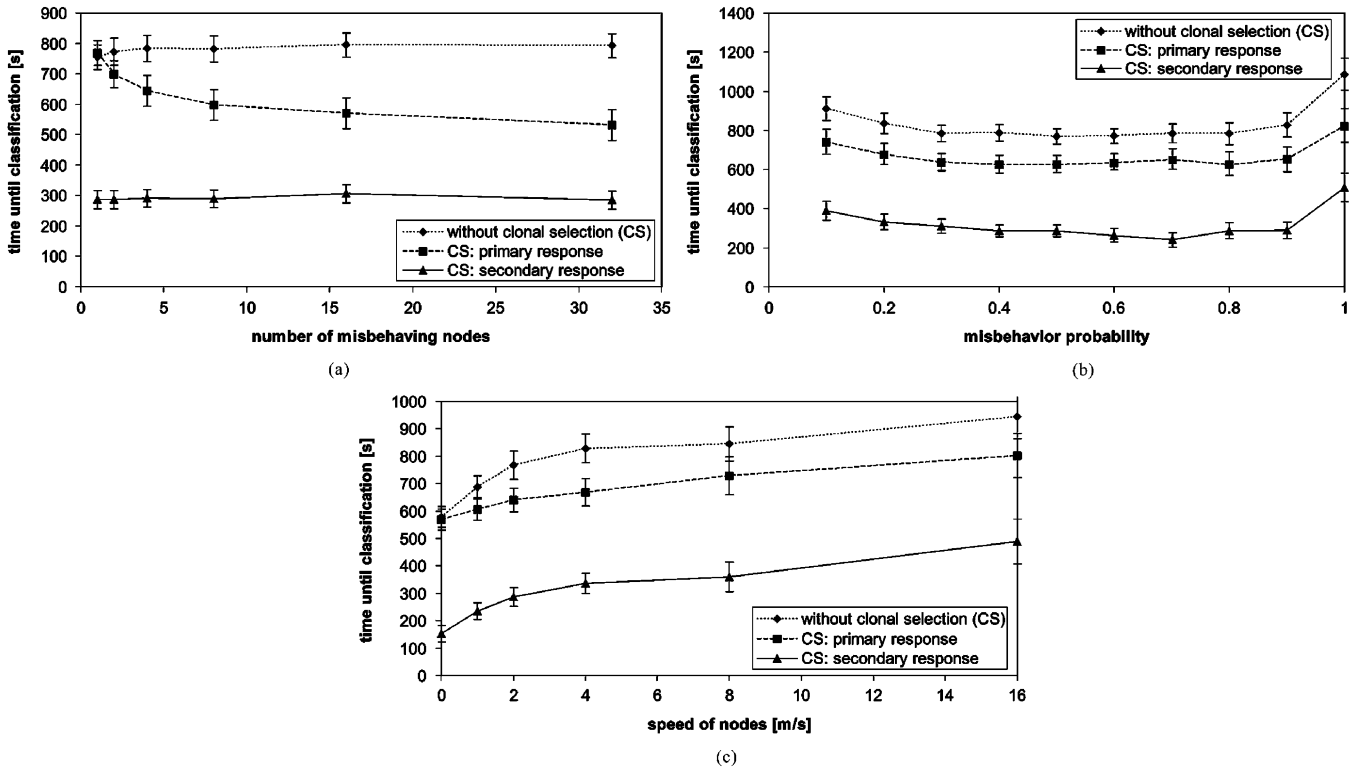


Fig. 3. Behavior of the detection system with parameters change. Time until true positive classification with respect to: (a) number of misbehaving nodes. (b) Misbehavior probability. (c) Speed of nodes. In all cases, target misclassification probability parameter α is set 0.0001. We do not put obtained false positive ratios on the graphs, because they vary in the very small range between 0.001 and 0.003.

metrics is by the parameter α, the classification error tolerance (Fig. 2). By decreasing the value of α, the false positive classification ratio decreases quickly to very small values, below the value 0.002 [Fig. 2(a)], causing a relatively small increase in the time needed for true positive classification [Fig. 2(b)]. This indicates that it is possible to choose α in the order of 0.0001 and obtain both a very small value of false classification ratio and a small time until classification.

One can notice from the Fig. 3(b) that the detection system has the largest time until classification when misbehavior probability is equal to 1. This comes from the fact that such a misbehaving node is visible as a neighbor only in the observation intervals in which it has its own traffic to send.

2) Impact of Clonal Selection: Clonal selection has a significant effect on the secondary response time, which is decreased by a factor 3–4 [Fig. 2(b)]. The false positive classification ratio is also slightly decreased [Fig. 2(b)]. Another positive effect of clonal selection is a slight decrease in primary response time (Figs. 2(b), 3). This slight decrease is more pronounced when the number of misbehaving nodes is large. This is because it is likely that a node that sees a specific misbehaving node for the first time was already exposed to other misbehaving nodes before. If there is only one misbehaving node, this effect does not occur [Fig. 3(a)].

3) Effect of Other Parameters: Fig. 3 shows that parameters other than α have a limited effect on the time until classification.

The effect of these parameters on the false positive classification ratio is even smaller. The values of obtained false positive classification ratios vary in the very small range between 0.001 and 0.003 (as mainly determined by the value of the parameter α ($\alpha = 0.0001$)) and this is the reason we do not show them on the graphs.

V. Conclusion

1) Results Obtained So Far: We have designed a method to apply an AIS approach for misbehavior detection in a mobile *ad hoc* network. Our simulations show good detection capabilities and the effectiveness of clonal selection at providing an accelerated secondary response. As explained in Section I-D, a quick secondary response is of special importance in the case of our protected system.

2) Lessons Learned and Future Work: In this early phase, it is premature to draw general purpose conclusions about the performance of the AIS approach. We need more experiments to extend our initial work to more misbehavior and traffic patterns. Instead, we would like to focus now on what the experience of designing this first phase tells us for the future.

3) Mapping IS to AIS: The most difficult problem we encountered was the mapping from the IS to the concrete problem of DSR misbehavior detection. We have followed the approaches in [7] and [10], but a large number of fundamental issues remain unclear. At the highest level, we still wonder what is the best choice for a target unit to be detected: the node, or sequence of messages, or a message itself. This choice could have an impact on the design challenges and possible use of the detection system. Even if we stay with the mapping we have designed here, questions remain vastly open.

The very definition of **genes** is one of them. We have defined them in an *ad hoc* way, trying to guess the definitions that would have the best detection capabilities. We made sure to have at least two correlated genes per misbehavior, in order to capture it efficiently. Indeed, we propose to use the correlation between genes as an important criterion in selecting the genes defined in the antigen structure. In a next phase, we are planning to automate the process of selecting genes. Correlation between genes from an offered set of genes can be computed from experimental data in a normal operation of the network without misbehaving nodes. Good pairs, triples or k-tuples of genes, which score high cross correlation, can be selected automatically. The final selection of candidate genes would still require to be screened by a human expert intervention, but this would be considerably simpler than designing genes from scratch, as we did. One can observe that such a gene selection process is not part of the natural IS, but one can view it as an accelerated *selection* process.

Though we targeted only the two mentioned types of misbehavior while selecting the genes for the antigen representation, the system should be able to detect more misbehavior types. For example, it should detect the denial of the service attack composed of too many route requests packets. We expect that a larger number of (automatically generated) genes would result in an antigen representation that enables for the detection of a larger number of different types of misbehavior.

An alternative is to define genes as arbitrary low-level bit patterns, and let negative and clonal selection do the job of keeping only the relevant antibodies. This would be more in line with the original motivation for using an AIS. A problem with this approach is that it would require many genes, and the processing effort needed to generate good detectors increases exponentially with the number of genes. A possible angle of attack is based on the observation that the IS is also a resource management system. Indeed, the IS has mechanisms to multiply IS cells and send them to the parts of the body under attack, thus, mobilizing resources where and when needed. An analogy here would be to use randomization: in a steady state, only a small, random subset of protocol event sequences is used to create antigens. When an attack is on (signalled by a danger signal, see the following), more events are analyzed in the regions that are in danger. In another direction, the response against a specific attack can depend on the intensity of the detection [23].

4) Innate System and Danger Signal: The model we implemented in this first step could be fortified with the additional mechanisms of the danger signals [14], i.e., the signal that controls activation of helper T-cells [the danger signal (Section II-B)]. Danger signals could be defined as network performance indicators (packet loss, delay). In the natural IS, the danger signal is intimately linked to the innate system. Here, the innate system could be mapped to the traditional approach, i.e., a set of predefined detection mechanisms as we developed in [4]. It is likely that many new attacks will be accompanied with symptoms that are not new. Thus, the innate system could be used as a source of the danger signal as well. This would free resources to focus the adaptive IS on the detection of truly new, unexpected misbehavior.

5) Virtual Protected Environment: The protected environment that we assume during the initial learning phase in our simulation might be impossible to provide in a real network. Also, there may be a need for some nodes to join the network later, when misbehavior is not surely absent. These nodes should still be able to learn normal behavior examples needed for negative selection. We propose and test a solution for this problem in our related paper [6]. The main idea is that the absence of a related danger signal is a good indicator that the antigen does not cause damage to the protected system. Such antigens are collected to form a virtual protected environment and represent the current self of the protected system.

6) Memory: We test the secondary response of our system by repeating the experiment (and misbehavior) but using the detectors created in clonal selection during the primary response. A way to reuse the good detectors created by clonal selection in a continually operating AIS is to promote some of them into memory detectors [6], [12]. Memory detectors are long lived and provide a secondary response to the misbehavior that is repeated after a long period of time.

7) Regulation: The translation of nonself antigen matching to misbehavior detection is done in a classical, statistical way. This should be compared to the regulation of B-cell and T-cell clonal division [16], which is algorithmically very different.

8) Parameter Tuning: Even if an appropriate mapping of IS to AIS is found, it remains that the performance is very sensitive to some parameters; the parameters have to be carefully tuned. It is unclear today whether this dependency exists in the IS, and if natural selection takes care of choosing good values, or whether there are inherently stable control mechanisms in the IS that make accurate tuning less important. Understanding this is key to designing not only an AIS as we do here, but also a large class of controlled systems.

APPENDIX I
DERIVATION OF (2)

We model the outcome of the behavior of a node as a random generator, such that with unknown but fixed probability θ a data set is interpreted as suspicious. We assume the outcome of this fictitious generator is iid. We use a classical hypothesis framework. The null hypothesis is $\theta \leq \theta_{\max}$, i.e., the node behaves well. The maximum likelihood ratio test has a rejection region of the form $\{M_n > K(n)\}$ for some function $K(n)$. The function $K(n)$ is found by the type-I error probability condition: $\mathbb{P}\{M_n > K(n)\} \,|\, \theta) \leq \alpha$, for all $\theta \leq \theta_{\max}$, thus, the best $K(n)$ is obtained by solving the equation

$$\mathbb{P}(\{M_n > K(n)\} \,|\, \theta = \theta_{\max}) = \alpha.$$

The distribution of M_n is binomial, with parameters n and θ, which gives the first part of the (2). For $n\theta > 5$, the distribution of M_n is well approximated by a normal distribution with mean $\mu = n\theta$ and variance $n\theta(1-\theta)$. After some algebra this gives $K(n) = \sqrt{n}\xi\sqrt{\theta_{\max}(1-\theta_{\max})} + n\theta_{\max}$.

APPENDIX II
LOOKUP-TABLE EXAMPLE

TABLE II
LOOKUP TABLE FOR $\alpha = 0.0001$

n	K(n)	n	K(n)	n	K(n)	n	K(n)
1	1	22	7	43	10	64	12
2	2	23	7	44	10	65	13
3	3	24	7	45	10	66	13
4	3	25	7	46	10	67	13
5	3	26	7	47	10	68	13
6	4	27	8	48	11	69	13
7	4	28	8	49	11	70	13
8	4	29	8	50	11	71	13
9	4	30	8	51	11	72	13
10	5	31	8	52	11	73	13
11	5	32	8	53	11	74	14
12	5	33	9	54	11	75	14
13	5	34	9	55	11	76	14
14	5	35	9	56	12	77	14
15	6	36	9	57	12	78	14
16	6	37	9	58	12	79	14
17	6	38	9	59	12	80	14
18	6	39	9	60	12	81	14
19	6	40	9	61	12	82	15
20	7	41	10	62	12	83	15
21	7	42	10	63	12	-	-

APPENDIX III
DETAILED ALGORITHM

Here is the detailed detection and classification algorithm that is executed in a node

```
//clonal selection may be enabled or disabled
//detecting may be used only if learning phase is already
done
isClonalSelEnabled=readIsCSEnabled();
phase=readPhase();//(by default LEARNING)
switch(phase){
 case LEARNING{
  phaseTimer=maximalLearningTime;
  numberOfCollectedDataSets=0;
  collectingDataSetsTimer=0;
  SetOfDetectors=
  CreateAnEmptyOrEmptyItIfExistsSetOfDetectors();
  numberOfDetectors=0;
  while(phase==LEARNING){
   if(aPacketSentReceivedOrOverheared){
   createOrUpdateNeighborsList();
    if(collectingDataSetsTimer==0 &&
       numberOfCollectedDataSets<maxNumOfCSD){
     openNewCollectingDataSets();
     collectingDataSetsTimer==deltaT;
    }//end if
    UpdateCurrentDataSetsIfTheyAreOpen();
   }//end if
   if(collectingDataSetsTimer==0){
    closeCurrentCollectingDataSets();
   }//end if
   if(phase!=LEARNING) break;
   if(phaseTimer==0){
    //create detectors by negative selection
    setOfSelfAntigens=
    createSelfAntigensFromCollectedDataSets();
    while(numberOfDetectors<TargetedNumDet){
     generateANewDetectorByRandom();
     if(isNewDetMatchingAnySelfAntigen()){
      deleteNewDetector();
     }//end if
     else{
      addNewDetectorToSetOfDetectors();
      numberOfDetectors++;
     }//end if else
    }//end while
    phase=DETECTINGandCLASSIFYING;
   }//end if
  }//end while
  break;
 }//end case
 case DETECTINGandCLASSIFYING{
  //with clonal selection as option
  if(!isSetOfDetectorsExists()){
   phase=WRONGuSE
   break;
  }//end if
```

```
//previous line prevents entering this phase
//in case that learning is never done
deleteScoresOfDetectorsIfTheyExist();
deleteDetectionDataForObservedNodesIfTheyExists();
while(phase==DETECTINGandCLASSIFYING){
 if(isPacketSentReceivedOrOverheared()){
  createOrUpdateNeighborsList();
  if(collectingDataSetsTimer==0 &&
   numberOfCollectedDataSets<maxNumOfCSD){
   openNewCollectingDataSets();
   collectingDataSetsTimer==deltaT;
  }//end if
  UpdateCurrentDataSetsIfTheyAreOpen();
 }//end if
 if(phase!=DETECTINGandCLASSIFYING) break;
 if(collectingDataSetsTimer==0){
  //do detection and classification
  createAntigensFromCurrentDataSets();
  deleteCurrentDataSets();
  checkAntigensFromLastDeltaTByDetectors();
  updateListOfDetectedNodes();
  updateDetectionResultsForObservedNodes();
  updateDetectionScoresForDetectors();
  deleteAntigensFromLastDeltaT();
  checkDetectionResultsForClassification();
  if(areNewNodesClassifiedAsMisbehaving()){
   if(isClonalSelEnabled){
    //do clonal selection
    MisbNodes=
    setOfNodesClassifedAsMisbInLastDeltaT();
    N=enumerateElementsOf(MisbNodes);
    for(i=1;i<=N;i++){
     misbNode=takeElement#i(MisbNodes,i);
     GoodScored=
     findGoodScoredForNode(misbNode);
     BedScored=
     findBedScoredForNode(misbNode);
     M=enumerateElementsOf(GoodScored);
     for (j=1;j<=M;j++){
      tmpDet=takeElement#j(GoodScored,j);
      oldDet=takeElement#j(BadScored,j);
      success=FALSE;
      while(success=FALSE){
       clone=makeClone(tmpDet);
       mutateClone(clone);
       isAnyMatching=
       isMatching(clone,SetOfSelfAntigens);
       if(isMatching)deleteNewClone();
       else{
        replaceOldByClone(oldDet,clone);
        success=TRUE;
       }//end if else
      }//end while
     }//end for
    }//end for
   }//end if
  }//end if
 }//end if
  }//end while
  break;
 }//end CASE
 case WRONGuSE{
 notifyWrongUse();
 }//end case
}//end switch
```

REFERENCES

[1] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. MOBICOM '00*, 2000, pp. 255–265.

[2] S. Buchegger and J.-Y. Le Boudec, "A robust reputation system for mobile ad hoc networks,", Lausanne, Switzerland, Tech. Rep. IC/2003/50, EPFL-DI-ICA, Jul. 2003.

[3] ——, "Performance analysis of the CONFIDANT protocol: Cooperation of nodes—Fairness in distributed ad hoc networks," in *Proc. IEEE/ACM Symp. Mobile Ad hoc Networking and Computing (MobiHOC)*, Lausanne, Switzerland, Jun. 2002, pp. 80–91.

[4] ——, "The effect of rumor spreading in reputation systems for mobile ad hoc networks," in *Proc. WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Sophia-Antipolis, France, Mar. 2003, pp. 131–140.

[5] J. Y. Le Boudec and S. Sarafijanovic, "An artificial immune system approach to misbehavior detection in mobile ad hoc networks,", Lausanne, Switzerland, Tech. Rep. IC/2003/59, EPFL-DI-ICA, Sep. 2003.

[6] S. Sarafijanovic and J. Y. Le Boudec, "An artificial immune system for misbehavior detection in mobile ad hoc networks with virtual thymus, clustering, danger signal and memory detectors," in *Proc. ICARIS'04 3rd Int. Conf. Artificial Immune Systems*, Catania, Italy, Sep. 2004, pp. 342–356.

[7] S. A. Hofmeyr, "An immunological model of distributed detection and it's application to computer security," Ph.D. dissertation, Dept. Comput. Sci., Univ. New Mexico, Apr. 1999.

[8] S. A. Hofmeyr and S. Forrest, "Architecture for an artificial immune system," *Evolut. Computat.*, vol. 7, no. 1, pp. 45–68, 2000.

[9] J. Kim and P. J. Bentley, "Evaluating negative selection in an artificial immune system for network intrusion detection," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO'01)*, San Francisco, CA, Jul. 7–11, pp. 1330–1337.

[10] ——, "The artificial immune system for network intrusion detection: An investigation of clonal selection with negative selection operator," *Proc. Cong. Evolutionary Computation (CEC'01)*, pp. 1244–1252, May 27–30.

[11] ——, "Toward an artificial immune system for network intrusion detection: An investigation of dynamic clonal selection," *Proc. Congr. Evolutionary Computation (CEC'02)*, pp. 1015–1020, May 12–17, 2002.

[12] ——, "Immune memory in the dynamic clonal selection algorithm," in *Proc. 1st Int. Conf. Artificial Immune Systems (ICARIS)*, Canterbury, Sep. 9–11, 2002, pp. 57–65.

[13] P. Matzinger, "Tolerance, danger and the extended family," *Annu. Rev. Immunology*, vol. 12, pp. 991–1045, 1994.

[14] P. Matzinger, "The danger model in it's historical contex," *Scandinavian J. Immunology*, vol. 54, pp. 4–9, 2001.

[15] L. M. Sompayrac, *How the Immune System Works*, 2nd ed. Oxford, U.K.: Blackwell, 2003.

[16] T. W. Mak, "Order from disorder sprung: Recognition and regulation in the immune system," *Phil. Trans. R. Soc. Lond. A*, vol. 361, pp. 1235–1250, May 2003.

[17] X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim: A library for parallel simulation of large scale wireless networks," in *Proc. 12th Workshop on Parallel and Distributed Simulations (PDAS'98)*, Banff, AB, Canada, May 26–29, 1998, pp. 154–161.

[18] D. B. Johnson and D. A. Maltz, "The dynamic source routing protocol for mobile ad hoc networks," *Internet draft, Mobile Ad Hoc Network (MANET) Working Group*, Feb. 2003.

[19] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," in *Proc. IMW'02*, Marseille, France, Nov. 2002, pp. 237–242.

[20] L. N. De Castro and F. J. Von Zuben, "Artificial immune systems: Part I basic theory and applications,", Tech. Rep. RT DCA 01/99, 1999.

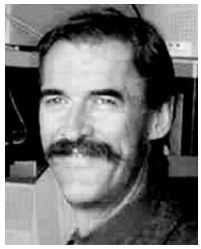
[21] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*. Berlin, Germany: Springer-Verlag, 2002.

[22] D. Dasgupta and F. González, "An immunity-based technique to characterize intrusions in computer networks," *IEEE Trans. Evol. Comput.*, vol. 6, no. 9, pp. 1081–1088, Jun. 2002.

[23] A. Somayaji and S. Forrest, "Automated Response Using System-Call Delays," in *Proc. 9th USENIX Security Symp.*, Denver, CO, Aug. 2000, pp. 185–197.

Slavisa Sarafijanović received the B.S. degree in electrical engineering from the School of Electrical Engineering, University of Belgrade, Serbia, in 2000. He is currently working toward the Ph.D. degree at EPFL, Lausanne, Switzerland.

He is participating in the Swiss NCCR project terminodes. org. His interests are in protection of computer networks and systems, i.e., making them immune to nodes and users misbehavior. This includes routing misbheavior, spam, and viruses.



Jean-Yves Le Boudec (F'04) graduated from Ecole Normale Superieure de Saint-Cloud, Paris, France and received the aggregation degree in mathematics and the doctorate degree from the University of Rennes, France, in 1980 and 1984, respectively.

In 1987, he joined Bell Northern Research, Ottawa, Canada, as a Member of the Scientific Staff in the Network and Product Traffic Design Department. In 1988, he joined the IBM Zurich Research Laboratory where he was Manager of the Customer Premises Network group. In 1994, he became a Professor at EPFL, where he is now a Full Professor and Head of the Institute for Communication Systems. He is a coauthor of the book *Network Calculus* (New York: Springer-Verlag, 2001). His interests are in the architecture and performance of communication systems.

Dr. Le Boudec won the IEEE Infocom 2005 Best Paper award (with Milan Vojnovic) for his work on the simulation of mobility models.